International Journal of Information Security (2022) 21:1349–1359
https://doi.org/10.1007/s10207-022-00614-6

REGULAR CONTRIBUTION

PSO + FL = PAASO: particle swarm optimization + federated learning = privacy-aware agent swarm optimization

Vicenç Torra[1,2] · Edgar Galván[3] · Guillermo Navarro-Arribas[4]

Accepted: 1 September 2022 / Published online: 22 September 2022

Abstract
In this paper, we present an unified framework that encompasses both particle swarm optimization (PSO) and federated learning (FL). This unified framework shows that we can understand both PSO and FL in terms of a function to be optimized by a set of agents but in which agents have different privacy requirements. PSO is the most relaxed case, and FL considers slightly stronger constraints. Even stronger privacy requirements can be considered which will lead to still stronger privacy-preserving solutions. Differentially private solutions as well as local differential privacy/reidentification privacy for agents opinions are the additional privacy models to be considered. In this paper, we discuss this framework and the different privacy-related alternatives. We present experiments that show how the additional privacy requirements degrade the results of the system. To that end, we consider optimization problems compatible with both PSO and FL.

Keywords Particle swarm optimization · Federated learning · Differential privacy · Masking · Differentially private social choice

1 Introduction

Privacy is a matter of trust. An agent can conceal some information when she does not trust the recipient of a message, or medium in which this message is transmitted. When the agent trusts both the medium and the recipient, the quality of the information being transmitted is higher [16] and the message will be more on line of agent's beliefs.

Federated learning [10] computes machine learning models without accessing the data of individual agents. In its basic form, federated learning considers a centralized entity that through a set of iterations computes the model, distributes the model back to the agents, and then, the agents send to the centralized entity the difference between their local models and the global one. One of the advantages of this approach is that is more privacy friendly than sending all agent's data to the centralized entity for it to compute the model.

Unfortunately, this privacy-based "guaranteed" federated learning is an illusion. The fact that agents only send summaries or differences between the central model and their own model can be rather sensitive [10,18]. In addition, the final central model can also lead to disclosure [1]. Computing a differentially private central model can be a solution for the latter, but not to the former problem. Multi-party computation can be seen as a solution of the former problem. In this case, agents need to trust the central entity, the designers of the protocol, as well as that there is no coalition between this entity and the agents, or a large enough coalition of agents, to make disclosure possible.

Thus, from a privacy perspective, agents need to trust all the other agents as well as the centralized entity, or they need to reduce both their communications and the quality of the information they supply. Additional privacy models [4,8,13,15] and technology are needed for this purpose.

✉ Vicenç Torra
vtorra@ieee.org

Edgar Galván
Edgar.Galvan@mu.ie

Guillermo Navarro-Arribas
guillermo.navarro@uab.cat

1 Department of Computing Science, Umeå University, Umeå, Sweden

2 School of Informatics, Skövde University, Skövde, Sweden

3 Naturally Inspired Computation Research Group, Department of Computer Science, Hamilton Institute, Maynooth University, Maynooth, Ireland

4 Department Information and Communications Engineering – CYBERCAT, Universitat Autònoma de Barcelona, Bellaterra, Catalonia, Spain

In this paper, we consider a unified framework for particle swarm optimization (PSO) [9] and federated learning, as a kind of continuum between different privacy levels. Recall that in PSO we have a set of agents each with its own position in the space of solutions, and their global goal is to jointly find an optimal solution. All share their positions, and the best of their positions is stored and shared, and is the basis of how each agent updates its own position. In our approach, we range from the situation in which agents do not mind to share all the information (i.e., PSO) to a strong privacy-preserving PSO-like environment aligned with federated learning. In this latter scenario, agents do not share their position, and the information supplied is locally protected and the final solution is differentially private.

Our analysis and experiments are in a two-dimensional setting. That is, all agents are located in a two-dimensional region. The goal is to minimize an optimization function in $\mathbb{R}$. We consider an homogeneous scenario where all agents have the same behavior. All use the same rules for updating their position and for deciding what and when information is transmitted. In heterogeneous environments, different agents will have different privacy preferences. That is, different agents will apply different strategies in their communication with a central authority. We can represent this situation in our scenario. It corresponds to a mixture of strategies. Agents can then reason on what and when to disclose i.e. their own strategy.

This paper introduces and evaluates privacy-aware swarm optimization (PAASO) in the context of federated learning (FL). To this end, we consider and evaluate different privacy alternatives within this framework and study how they affect the PSO optimization problem. Our results show that, in general, the use of privacy mechanisms does not avoid convergence although they make it slower.

The structure of the paper is as follows. In Sect. 2, we review some definitions related to data privacy. In Sect. 3, we introduce our approach. In Sect. 4, we describe our experiments and results. Finally, Sect. 5 draws some conclusions.

2 Some privacy definitions

We consider privacy from different perspectives or dimensions. In general, we would like to avoid the disclosure of private data and information in all possible stages of the federated learning process. This includes the computations that need to be performed, the final and partial outputs generated by the process, and the data exchanged between the agents.

We rely in the well-known model of differential privacy [5,6]. The classical definition of differential privacy considers two databases D_1, D_2 that differ only in one record (denoted as $d(D_1, D_2) = 1$) and an interactive scenario where the user sends queries to the database.

Definition 1 A randomized query function $\mathcal{K}$ gives ϵ-differential privacy if for all $S \subseteq Range(\mathcal{K})$, and for all D_1, D_2 such that $d(D_1, D_2) = 1$,

$$Pr[\mathcal{K}(D_1) \in S] \leq e^{\epsilon} \times Pr[\mathcal{K}(D_2) \in S] \quad (1)$$

Broadly speaking, it states that the addition or deletion of a single record should not be noticeable from the answers to the query $\mathcal{K}$ up to a given bound determined by ϵ. Lower values of ϵ provide more privacy, usually and as expected, at expenses of higher information loss.

We usually assume that the randomization is performed by a trusted data curator which has access to all the database. An slightly different definition of differential privacy might be more interesting in a federated learning environment, where each agent generates its own, possibly protected, data. In this case, differential privacy can be considered from a local point of view giving up to the so called ϵ-*local differential privacy*. The difference is given by the fact that each user randomizes its own data which is then stored in the database. The definition is analogous to differential privacy, but in the local case we consider that the randomized function $\mathcal{K}$ takes as input data from a single user instead of taking the whole database.

In our case, we consider the privacy that can be achieved by a particle swarm optimization in a federated learning scenario. To provide a broad privacy analysis, we can establish different disclosure dimensions in the proposed framework. Information can be disclosed in several stages as described next.

2.1 Avoiding disclosure from the computation

It is relatively common to rely on secure multiparty computation (SMC) to provide privacy preserving computations in federated learning scenarios [17]. Secure multiparty computation provides a safe tool for agents to jointly compute any desired function without revealing each agent input.

Formally, given a function to be computed jointly by several agents, a protocol is defined that specifies how the function is computed. In typical secure multiparty scenarios, the final result is then transferred to the agents. From a privacy point of view, no agent learns more than what can be inferred from its own data and the output of the computation. Partial computations and any information on the transmission do not lead to additional knowledge.

Honest but curious, and malicious intruders are the usual type of attacks considered. Honest but curious agents follow the protocol. They try to acquire knowledge by means of observing when the communications take place and analyzing any information being transmitted (even encrypted ones). Malicious intruders do not necessarily follow the protocol and can e.g. inject noise to transmissions, provide wrong

information, or just sent meaningless messages to acquire knowledge. Coalitions of agents are other types of attacks because, naturally, joining their efforts a set of agents can be able to acquire information on other ones.

It might seem that SMC techniques in general introduced important penalties both in terms of efficiency and communication overhead. Despite this initial impression, currently there are relatively efficient general solutions for secure multiparty computation [2], and it has been successfully used in federated learning [3].

In our case, the computations to be performed are relatively simple and can be addressed by well-known SMC protocols. We will not get into detail about the actual tools used in order to focus on the description and scope of our proposal in disclosure from the output and communication's content and the privacy guaranties provided in terms of differential privacy.

2.2 Avoiding disclosure from the output

When several agents need to decide on a given output, e.g. based in a voting scheme, the whole procedure as well as the output can disclose information about the agents. In the case of PSO, as we will see, agents need to agree in a global best position, which could likely yield information about the actual position of some agents.

In [14], it is shown that a differentially private consensus can be achieved based on a random dictatorship voting scheme. The standard non-privacy preserving random dictatorship considers that a set of agents I vote on a given set of alternatives A, so each agent $i \in I$ has a defined preference $\succeq_i$ defined in terms of $A \times A$. An agent i is selected according to a uniform distribution on the whole set of agents I, and then, $\succeq_i$ is used to select the alternative outcome (i.e., agent i is randomly chosen as the dictatorship).

An alternative voting procedure that achieves differential privacy is defined in [14] as:

Definition 2 Let $A = \{a_1, \ldots, a_m\}$ be the set of alternatives. Then, given the set of agents I with the corresponding preference relations $\succeq_i$ for $i \in I$ on the alternatives A, enlarge I with a set of agents $I_0 = \{e_1, \ldots, e_m\}$ such that $\succeq_i$ for $i \in I_0$ has as its preferred alternative the ith alternative in A. Then, apply uniform random dictatorship on $I \cup I_0$.

In this last definition, it can be shown that differential privacy is achieved for any

$$\epsilon \geq \log \frac{2|I \cup I_0|}{|I \cup I_0| + 1} \tag{2}$$

2.3 Avoiding disclosure from communication's content

Agents need to communicate, for instance, in order to conduct the voting scheme described in the previous section. In such case, agents send their data, which as we will see is a *direction* based on their local position. Assuming agents want to keep their position private, the direction helps in concealing the position but might not be enough. In order to provide stronger privacy warranties, agents can mask the data they send.

PRAM (Post-RAndomization Method) [4,7] is a well-known statistical disclosure control technique, initially defined for categorical data. Given a set of categories $C = \{c_1, \ldots, c_c\}$, we define a $c \times c$ stochastic matrix (or Markov matrix) P so $\sum_{c_j \in C} P(c_i, c_j) = 1$. Then, the masking is produced by replacing each c_i for c_j with probability $P(c_i, c_j)$. When this technique is performed by the respondent, it is usually referred as randomized response (RR). The application by the respondent has a drawback. The respondent can only access its own data and protection is done in real time. Therefore, the stochastic matrix cannot be generated taking into account the distribution of the whole data set, but on our expectations of this distribution. In our case, agents will apply RR to the data they need to send (i.e., a discretized version of direction) in order to establish the global parameter of the PSO. Sections 3.4 and 3.3 will provide more detail on the use of PRAM in our proposal.

It is know that RR provides ϵ-local differential privacy if

$$e^{\epsilon} \geq \max_{c_i} \frac{\max_{c_j} P(c_i, c_j)}{\min_{c_j} P(c_i, c_j)} \tag{3}$$

This means that we can provide strong privacy requirement in the data locally released by agents.

We will apply PRAM where the matrix is defined in terms of a probability parameter (p_m). Then, given the value c_i, we have that $P(c_i, c_i) = p_m$ and $P(c_i, c_j) = (1 - p_m)/(c - 1)$. In other words, the probability that the masking does not modify a category is p_m. Then, with probability $1 - p_m$ the category will be replaced by any other. The selection of any other category follows a uniform distribution. Naturally, this definition applies to all i. Therefore, the Markov transition matrix has the value p_m in each element of the diagonal and $(1-p_m)/(c-1)$ in all other positions. If $p_m > (1-p_m)/(c-1)$, then $\epsilon \geq \ln(c-1)/(1-p_m)$.

3 An scenario

Our starting point is to observe particle swarm optimization (PSO) and see that it fits well in the federated learning (FL) framework. This is so because in both PSO and FL

we are optimizing a function that is defined by means of the behavior of a set of *independent* agents (i.e., the particles in PSO and different devices in FL). This observation permits to define a PSO à la FL. Then, these agents have privacy requirements. They are local ones, that can be implemented locally by each agent, and global ones, that need agreement. Taking these requirements into consideration, we define privacy-aware agent swarm optimization (PAASO).

3.1 No-privacy PSO

Let us start reviewing particle swarm optimization and consider an example. To make the case simple, we consider in our experiments functions in the two dimensional space. Nevertheless, our proposal is general. Then, each agent has its own position in the space and each agent knows the value of the function to be optimized on its current own position (and in previous positions as well, if the agent has not forgotten).

Standard PSO follows the following schema.

- **l1** Assign agents positions x_i to agents $i = 1, \ldots, s$;
- **l2** $p_i := x_i$; # Assign best agent position to x_i
- **l3** $g = \arg\min_{p_i} f(p_i)$; # g is the best known position
- **l4** Assign agents velocities v_i to agents $i = 1, \ldots, s$;
- **l5** iterate
  - **l5.1** for each agent $i = 1, \ldots, s$
    - **l5.1.1** (r_p, r_g):= random numbers, uniform in ([0, 1], [0, 1])
    - **l5.1.2** $v_i := \omega v_i + \phi_p r_p(p_i - x_i) + \phi_g r_g(g - x_i)$;
    - **l5.1.3** $x_i := x_i + v_i$;
    - **l5.1.4** if $(f(x_i) < f(p_i))$ then $p_i := x_i$;
    - **l5.1.5** if $(f(p_i) < f(g))$ then $g := p_i$; # p_i is shared
- **l6** until (end-condition);

We denote by x_i, v_i and p_i the position and velocity of the ith agent, and the best position found so far, respectively. Then, g is the best global position found so far. Function f denotes the function to be optimized (minimized in our study) and s the number of agents. Then, ω is the inertia weight, ϕ_p and ϕ_g are acceleration coefficients.

Lines l1–l4 correspond to initialization, and lines l.5.* are the iterative process to find the optimal solution to the problem. Lines l5.1.1–l5.1.3 are to establish the new velocity and position of the agent and lines l5.1.4 and l5.1.5 the updating of agent's best position (p_i) and of the global best position (g_P).

If we consider situations in which privacy is not an issue, any agent can send at any time any information about its current and past positions (i.e., x_i and p_i), as well as the value of the function evaluated at them. Therefore, we can follow without any problem the PSO protocol.

3.2 PSO lacking trust: PSO à la FL

When the agent trusts the central entity, and the multiparty computation protocol, the agent presumes that all the supplied information will not lead to a direct inference from the central entity i.e. that disclosure will not take place. Therefore, we can build a cryptographic protocol using secure multiparty computation that computes the minimum function from the values supplied by the agents at any time and stores the corresponding position if the evaluation is better than the last stored one.

Still, lack of trust may prevent an agent to send its position and its evaluation. Also, it is not only a lack of trust on the central entity and the multiparty computation model; it is also about the trust to other agents that matters. For example, a coalition of all but one agent can of course be used to determine the position of the one left out. Smaller coalitions can build estimations of other agents positions. This would correspond to honest but curious intruders. Malicious intruders and dynamic adversaries (that can fool the protocol and/or make others to also fool the protocol) may have greater advantage.

An approach more privacy-friendly and in line with federated learning would be that agents do not supply their own position but a direction between the current global best one and their own. Recall that in federated learning agents do not share data but gradients. In our context, that is, each agent supplies the direction $d_i = p_i - g$ as well as $f(p_i)$. This conceals p_i and avoids any unintentional disclosure of agent's position. To do so, we need to revise the previous PSO definition as g cannot be correctly updated. Instead, we introduce the *position* of the central entity which is now public together with its evaluation. We call this position p_G. This position is iteratively updated using the directions of the agents. A multiparty computation protocol would ensure that agent's directions are not disclosed; the objective function is locally computed by the central authority. Then, the central authority shares the global position and its evaluation (i.e., p_G and $f(p_G)$).

If all agents have as its goal to optimize the same function, then they will update all their positions in a similar way. Otherwise, if different agents have different objectives, different approaches can be used for moving to the next position. In the next algorithm, we consider the case that all agents want to optimize the same function.

- **l5** iterate
  - **l5.1** for each agent $i = 1, \ldots, s$

- **l5.1.1** (r_p, r_g):= random numbers uniform in ([0, 1], [0, 1])
- **l5.1.2**

$$v_i := \begin{cases} \omega v_i + \phi_g r_g (p_G - p_i) & \text{if} f(p_G) < f(p_i) \\ \omega v_i + \phi_p r_p (b_i - p_i) & \text{otherwise.} \end{cases}$$

- **l5.1.3** $x_i := x_i + v_i$;
- **l5.1.4** if $(f(x_i) < f(p_i))$ then $p_i := x_i$;
- **l5.1.5** v_i is transmitted

– **l5.2** $v := (1/s) \sum_i v_i$; # v_i is shared
– **l5.3** $p_G := p_G + \omega_G v$

– **l6** until (end-condition);

Nevertheless, also in line with what happens in federated learning: when we send only directions (subtractions), they can lead to disclosure. Agents (and coalitions of agents) can use these directions to learn the position of a particular agent. In addition, the final solution (i.e., the final outcome of the system with its position and evaluation) can lead to sensitive information. In particular, the solution will not be differentially private. Note that when a solution can be used to infer the participation (or absence) of an agent, then the solution is not differentially private. To see that the approach is non-differential privacy, we can consider an agent located in regions of large values of the objective function, while all other agents are in other regions.

3.3 A differentially private solution

We can solve the problem just reported by reducing the dimensionality of the space of agents' opinions (i.e., in this case the space of *directions* / *subtractions*) and proceeding with a differentially private aggregation procedure. More precisely, instead of allowing the agents to provide any direction, we will only allow for a few finite set of alternatives. In our example, we consider a discrete set of directions. For example, 4 possible directions would be: front, right, left, back. So, each agent computes its vote α_i and casts it. Then, instead of aggregation, *we consider voting*. A differentially private voting mechanism (as the one described in Definition 2) will provide privacy to agents. We use *dpv* to denote the differentially private voting mechanism, and a function *velocity* to transform the decision into a velocity. So, we replace lines l5.1.5 and l5.2 above by:

– **l5.1.5** $\alpha_i = vote(v_i)$ is transmitted; # agent casts vote
– **l5.2** $v := velocity(dpv(\alpha_i))$

To implement this approach, we consider the differentially private mechanism discussed in Sect. 2.2.

3.4 Agents mask their vote

The approach above still assumes that an agent trusts the entire system, the multiparty computation protocol, and that neither the central authority nor the other agents are sniffing the communications to learn their positions. If this is not the case, an agent can mask its vote using a masking method *mm*. This corresponds to apply the following:

– **l5.1.5** $\alpha_i = vote(mm(v_i))$ is transmitted; # agent casts vote

To implement this approach, we will use the masking procedure discussed in Sect. 2.3.

3.5 PAASO: Privacy-aware agent (swarm) optimization

As a summary, there are the following different protection mechanisms for the agent.

– An agent votes for a direction within a reduced set of possibilities.
– An agent may reduce additional need of communication if the vote is only sent when its own position is better than the best current one,
– An agent can conceal its real vote by means of a masking technique.

The last one is the most privacy-friendly for an agent. The first one reduces the information supplied, the second one reduces the information supplied still further, but in both cases the information supplied is correct. Differentially private voting makes the aggregation private, but the data supplied by agents are unprotected (except for SMC protocols). The third case introduces randomization (by means of data masking—local differential privacy). In this paper, we compare these approaches.

As a summary, from the lowest to the highest level of privacy guarantees. We have the following types of privacy-aware agents.

– Standard PSO (PSO). Agents inform of their position at any time, and the objective function for that position is shared or computed by the central authority.
– Federated learning-like PSO (FL). Agents inform of their direction toward a global/consensus position at any time and of the objective function for that position.
– Differentially private PSO (aDRD). Agents inform of their direction toward a global/consensus position using a differentially private mechanism. Agent's decision on whether or not to vote is made according to a probabil-

ity. Here, DRD stands for Differentially Private Random Dictatorship.
- Differentially private PSO voting only when better (bDRD). Similar to the previous case, but vote can only be casted if agent's solution is preferable. As in the previous case, agent's decision on whether or not to vote also depends on a probability.
- Differentially private and masked voting for PSO (PbDRD). As in the previous case, agents vote when they have a contribution to make but in this case they can obfuscate their opinion. The voting procedure is as in the previous case differentially private.

4 Experiments and analysis

In this section, we describe the experiments we have considered to evaluate our proposal. In general, as usual in the privacy literature, the more the privacy level, the less the quality of the solution. Although this is so in most of the cases, the literature also shows that in some situations and scenarios, the introduction of privacy constraints does not imply a significant reduction in the quality of the solutions. In fact, the literature on privacy for databases has shown that some perturbation on the original data to guarantee some privacy level can have no effect on data-driven machine learning models, and in some cases [11], the perturbation even increases the accuracy of the models. Unfortunately, this will not be the case in our experiments. We have loss of quality in terms of slowing convergence.

Notice that we have evaluated the utility of our proposal by considering the performance of the PSO optimization. This is a specific information loss (or utility) measure commonly used for computation an data-driven protection methods [13]. Other generic utility measures might not be so relevant in this context, where the computation to be performed is clearly defined.

In order to evaluate different privacy levels of agent (swarm) optimization, we have considered different optimization problems. The global goal of the system is to find the optimal solution. In addition, agents have also interest on finding this optimal solution and have as their *local* objective function the same objective function of the whole system.

In order to make analysis and computation simple, all agents have the same behavior. That is, all agents apply the same approach to update their own position and velocity. They all apply the same masking technique (if any) and vote using the same strategy (in case of voting). In this sense, the multiagent system can be seen as homogeneous. Naturally, more complex and heterogeneous scenarios can be considered. In particular, from a privacy point of view, it is relevant to consider more complex decisions on what information needs to be shared and when the information needs

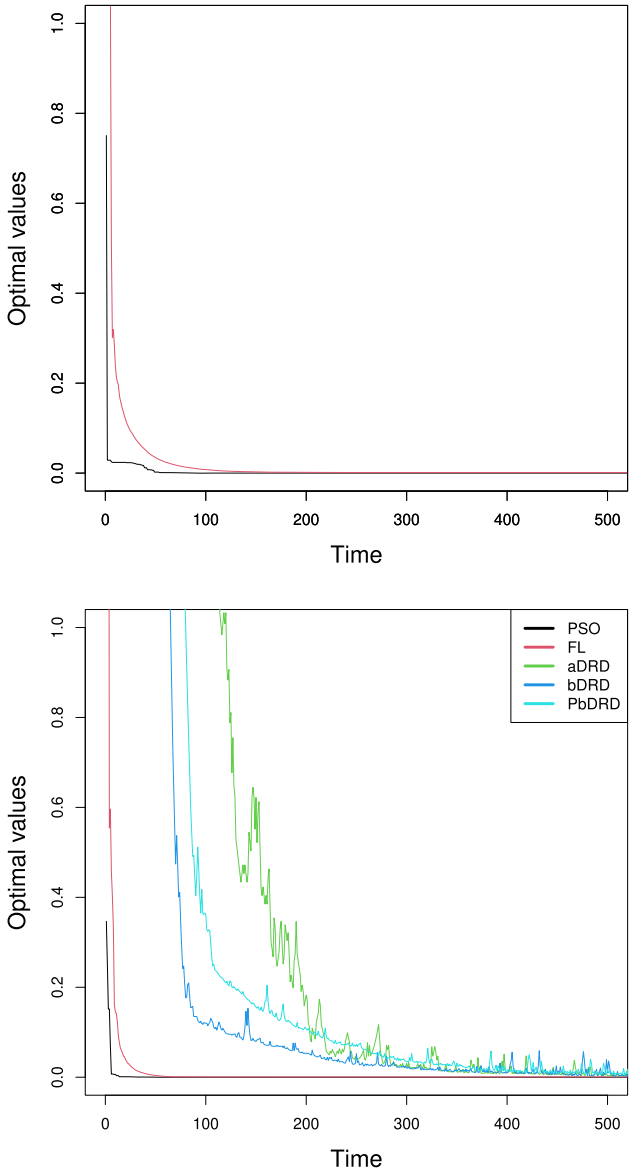

Fig. 1 Optimal values (y-axis) achieved for the different Agent Swarm Optimization approaches (PSO, FL, aDRD, bDRD, PbDRD) with respect to time (x-axis) in two different executions. The parameters are as follows: function f_4, number of voting alternatives $k_\alpha = 8$, 50 agents, $\omega = 4.00$, $\phi_p = \phi_g = 2.00$, $\omega_G = 0.005$, $p_c = 1.0$, and $p_m = 0.9$

to be considered. Also, different users have different privacy preferences and that needs to affect agent's decisions.

In the remaining part of this section, we describe the problems considered, the parameters of the system, and also the results obtained.

4.1 Problems considered

As explained above, we have considered functions with two inputs and evaluated in $\mathbb{R}$. That is, functions of the form

$f : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$. All functions are to be minimized. The eight problems considered are the following:

1. Quadratic function ($x_1, x_2 \in [-100.0, 100.0]$):

$$f_1(x_1, x_2) = x_1^2 + x_2^2$$

2. Schwefel's problem 2.22 ($x_1, x_2 \in [-10.0, 10.0]$):

$$f_2(x_1, x_2) = |x_1| + |x_2| + |x_1| \cdot |x_2|$$

3. Schwefel's problem 1.2 ($x_1, x_2 \in [-100.0, 100.0]$):

$$f_3(x_1, x_2) = x_1^2 + (x_1 + x_2)^2$$

4. Generalized Rosenbrock's function ($x_1, x_2 \in [-2.0, 2.0]$):

$$f_4(x_1, x_2) = 100 * (x_2 - x_1 * x_1)^2 + (x_1 - 1)^2$$

5. Generalized Schwefel's problem 2.26 ($x_1, x_2 \in [-500.0, 500.0]$):

$$f_5(x_1, x_2) = -x_1 \sin(\sqrt{|x_1|}) - x_2 \sin(\sqrt{|x_2|})$$

6. Rastrigin's function ($x_1, x_2 \in [-5.12, 5.12]$):

$$f_6(x_1, x_2) = 2 \cdot 10 + x_1^2 - 10\cos(2x_1\pi) + x_2^2 - 10\cos(2x_2\pi)$$

7. Ackley's function ($x_1, x_2 \in [-32.768, 32.768]$):

$$f_7(x_1, x_2) = -20e^{-0.2\sqrt{0.5(x_1^2+x_2^2)}} - e^{0.5\cos(2x_1\pi)+cos(2x_2\pi)} + 20 + e$$

8. Griewank function ($x_1, x_2 \in [-600.0, 600]$):

$$f_8(x_1, x_2) = 1 + (1/4000)(x_1^2 + x_2^2) - \cos(x_1) * \cos(x_2/\sqrt{2})$$

These problems are of different complexity. Details on these functions are given in the work by Sengupta et al. [12]. Function f_1 can be considered the simplest one while some others are multimodal with some level of deceptiveness. All optimal solutions have an objective function equal to zero, except for function f_5 that has its global minimum with a value of -12569.5. We have chosen this functions as they are commonly used unimodal and multimodal benchmark functions for PSO [12].

4.2 Agent (swarm) optimization

For implementing different types of privacy-aware agent optimization, we have considered different algorithms described in Sect. 3. We will denote them using the names in Sect. 3.5. That is, PSO (no privacy), FL (privacy of position), aDRD (directions shared, differentially private voting), bDRD (only voting direction in a better position, differentially private voting), PbDRD (PRAM-based masking + differentially private voting, if better). The implementation of the system is based on differentially private random dictatorship (as described in Definition 2) for the private voting, and PRAM for agent's individual privacy (and, thus, local differential privacy, Sect. 2.3).

The implementation of the system requires fixing a number of parameters. We have selected these parameters heuristically with the goal of having a generous range of parameters to find good solutions and also being computationally feasible. In particular, we have considered the number of agents to be 50, 100, and 200. Note that the more agents, the most computationally intensive the problem results. A different number of iterations have been considered, up to 1000. In the figures, shorter sequences of values are displayed.

As explained above, all agents have the same behavior and use the same strategy. Therefore, we need a single set of parameters for the whole system. In our case, this means that we need to establish ω (the inertia weight), ϕ_p and ϕ_g (the velocities), ω_G (the global inertia weight), k_α is the number of voting options (that is, it permits us to represent the angles $2\pi \cdot i/k_\alpha$ for $i = 0, \ldots, k_\alpha - 1$ from direction $(1, 0)$), p_c is the probability to cast a vote and p_m the probability associated with the masking method. Not all privacy-aware solutions use all parameters, as we have seen in the aforementioned definitions.

Then, we have considered for ω and ω_G the values $\{0.005, 0.001, 0.05, 0.1, 0.2, 0.4\}$, and we have considered $\phi_p = \phi_g$ and the values $\{0.01, 0.05, 0.1, 0.2, 0.5, 1.0, 2.0\}$. These set of values have been found heuristically. They have been selected to include some parameterizations that are suitable for PSO, other parameterizations that are suitable for FL, and other good for other type of privacy-aware solutions. In relation to k_α, we have used $k_\alpha = 8$ in most of the experiments. We have also used $k_\alpha = 4$, 16, and 32. The masking method used is PRAM. Then, the probabilities p_c and p_m have been selected among the values $\{0.6, 0.7, 0.8, 0.9, 1.0\}$. Naturally, when $p_c = 1$, the agent always communicates. In addition, a probability of $p_m = 1$ in PRAM means no protection. From a differential privacy perspective, these values of p_m different to one correspond to $\epsilon = 2.35, 2.79, 3.33, 4.14$.

For each configuration of values, we performed 20 independent runs. Each run with 1000 iterations and 50 agents has taken 0.033 seconds, and thus, a configuration takes 0.66 seconds.

4.3 Discussion of results

As we have mentioned above, we have considered different types of privacy-aware agents, that is, from the PSO solution in which agents share all information to the case of agents masking their votes and considering a differentially private voting mechanism with only a few alternatives.

As a first illustration, we show different executions of problem f_4. Figure 1 (top and bottom) displays the solutions of PSO, FL, $aDRD$, $bDRD$, and $PbDRD$ with $p_m = 0.9$ for two different executions. The other parameters correspond to $k_\alpha = 8$, $\omega = 4.00$, $\phi_p = \phi_g = 2.00$, and $\omega_G = 0.005$. This figure shows this convergence for all methods after 500 iterations. We see the fastest convergence for PSO and the slowest for the most privacy-friendly one ($PbDRD$ with $p_m = 0.9$). Convergence for the federated learning FL solution is slower than the one of PSO but faster than the others. The two alternatives with a differential privacy voting strategy are similar. Comparing the two figures, we can see the variability of the solution.

Due to this variability of the solution in different executions, Fig. 2 displays the mean objective function achieved for the 20 different executions considered for the same problem f_4 (using the same parameters as above). This figure clearly shows the difference of convergence for the three alternative approaches FL, $aDRD$, and $bDRD$.

Figure 3 displays mean results for the $PbDRD$ with different values of the parameter p_m considered. In particular, we have considered $p_m = \{0.6, 0.7, 0.8, 0.9, 1.0\}$. The standard deviations of the objective function in the last iteration for each of the p_m range between 0.03 and 0.05. From the figure, we get the impression that there are no fundamental differences on the parameter used for masking the own data. That is, that a strong protection (i.e., with $p_m = 0.6$) and no protection at all ($p_m = 1.0$) gives mainly the same results. This is not completely true as a change of scale would show that the convergence is a little bit slower when the protection is higher. Nevertheless, the trend is similar for all values of p_m. A consequence of this result is that the use of a masking method / local differential privacy by the agent does not have a major influence on the outcome if the final result is obtained using a differentially private voting mechanism. We discuss the need of applying this masking in Sect. 5.1.

In general, all different privacy-aware strategies converge to the desired solution. What differs is the speed of convergence. Among the problems, f_5 is the most difficult and the optimal solution found with any voting mechanism is far from the optimum.

PSO is the one with the fastest convergence for the problems considered. FL is next, it has a slower convergence, but it is quite similar. Other privacy strategies have significantly slower convergence.

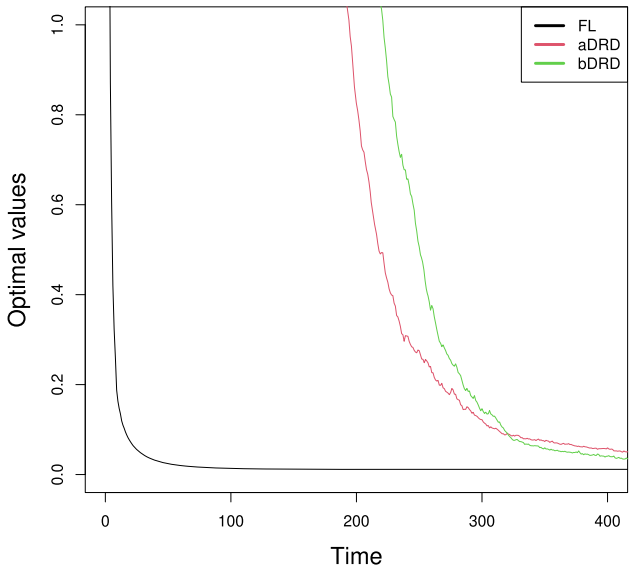

Fig. 2 Mean objective function for 20 executions for FL, aDRD, and bDRD. Function f_4, number of voting alternatives $k_\alpha = 8$, 50 agents, $\omega = 4.00$, $\phi_p = \phi_g = 2.00$, and $\omega_G = 0.005$. We use $p_c = 1.0$

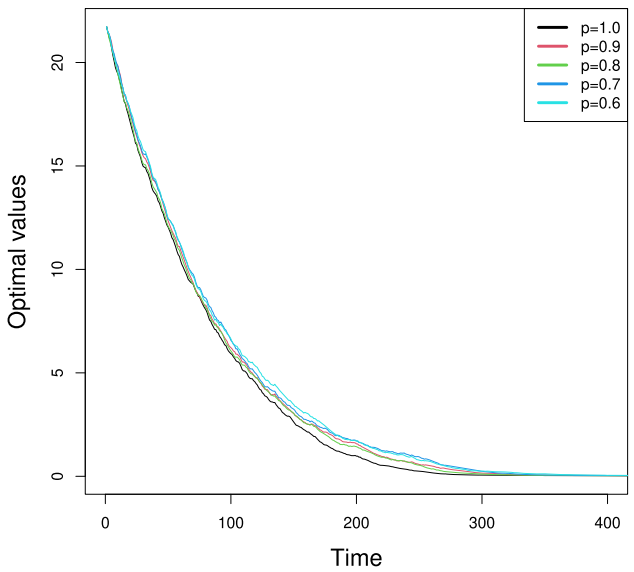

Fig. 3 Mean objective function for 20 executions for $PbDRD$ and different values of the parameter p_m. Function f_4, number of voting alternatives $k_\alpha = 8$, 50 agents, $\omega = 4.00$, $\phi_p = \phi_g = 2.00$, and $\omega_G = 0.005$. We use $p_c = 1.0$

This is a general rule, but different parameterizations have different effects and some are not suitable to e.g. achieve the best optimal solution e.g. in Fig. 4 we see that for parameters $\omega = 0.005$, $\phi_p = \phi_g = 2.00$, and $\omega_G = 0.01$ the mean performance of FL is worse than the other privacy-aware strategies for the same parameters. In fact, a close look to the 20 executions of FL with these parameterizations shows that convergence almost never takes place. The standard devia-

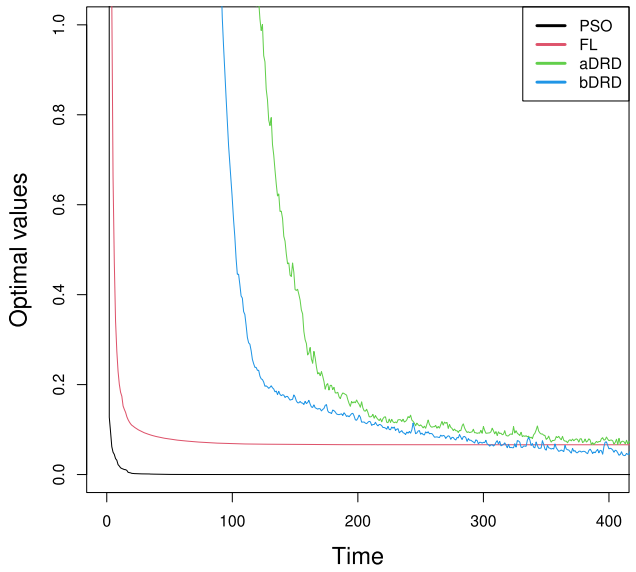

Fig. 4 Mean objective function for 20 executions for PSO, FL, $aDRD$, $bDRD$. Function f_4, number of voting alternatives $k_\alpha = 8, 50$ agents, $\omega = 0.005$, $\phi_p = \phi_g = 2.00$, and $\omega_G = 0.01$. We use $p_c = 1.0$

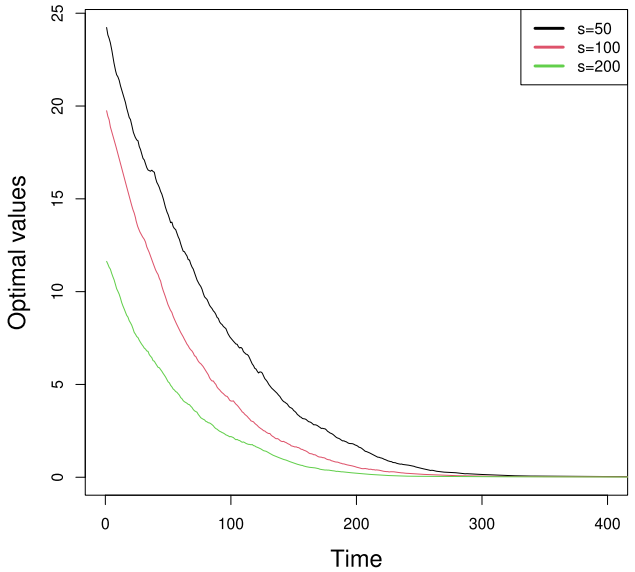

Fig. 5 Mean objective function for 20 executions for bDRD. Function f_4, number of voting alternatives $k_\alpha = 8$, $\omega = 0.005$, $\phi_p = \phi_g = 2.00$, and $\omega_G = 0.01$. We use $p_c = 1.0$. The figure illustrates different number of agents $s = 50, 100, 200$

tions of the objective function at the last iteration of each of the curves range between 0 and 0.030.

We have also compared different values for k_α for several combinations of the other parameters. In particular, we have considered the following values 4, 8, 16, 32 for k_α. For these values, there is no significant difference in the results. In some cases, even larger options give a slightly worse convergence. A reason for this somehow unexpected behavior can be the differentially private voting mechanism. The more possible options, the more *noise* is added by the mechanism. The *noise* is, roughly speaking, linear on the set of options (see Definition 2).

What has a clear effect on the convergence is the number of agents considered. Most experiments reported here consider a set of 50 agents. Figure 5 shows the results of a problem when the number of agents is $s = 50$, 100, and $s = 200$, and we clearly see a quicker convergence each time we double the size of the set of agents.

To conclude this section, we represent in Fig. 6 some of the configurations that achieve the best convergences for the problems f_1–f_8 considered. The figure represents only the first 400 iterations. As stated above, the solution for f_5 is far from the optimum. For all privacy-aware solutions, we get a minimum value of -7.9, while for PSO we get -837 and as stated above the optimal solution has an objective function of -12569.5.

5 Conclusions

The results of our experiments discussed in the previous section can be summarized in the following terms.

- In general, the use of privacy mechanisms does not avoid convergence. These mechanisms make convergence slower.
- The most effective way in terms of converge is PSO and then FL.
- The use of additional mechanisms as reducing the number of options to a few (i.e., the parameter k_α) and differential privacy voting makes convergence more difficult.
- Additional privacy mechanisms as local protection by means of PRAM do not seem to have a strong effect in the convergence.

In relation to the parameters of the system.

- The exact value of k_α has low effect on the convergence.
- The number of agents in the system is a key factor in the convergence. We have seen that doubling the number of agents can speed the convergence up significantly.
- Optimal values for parameters ω, ϕ_p, ϕ_g, and ω_G depend on the problem and the type of privacy strategy. Therefore, they need to be optimized for each architecture.

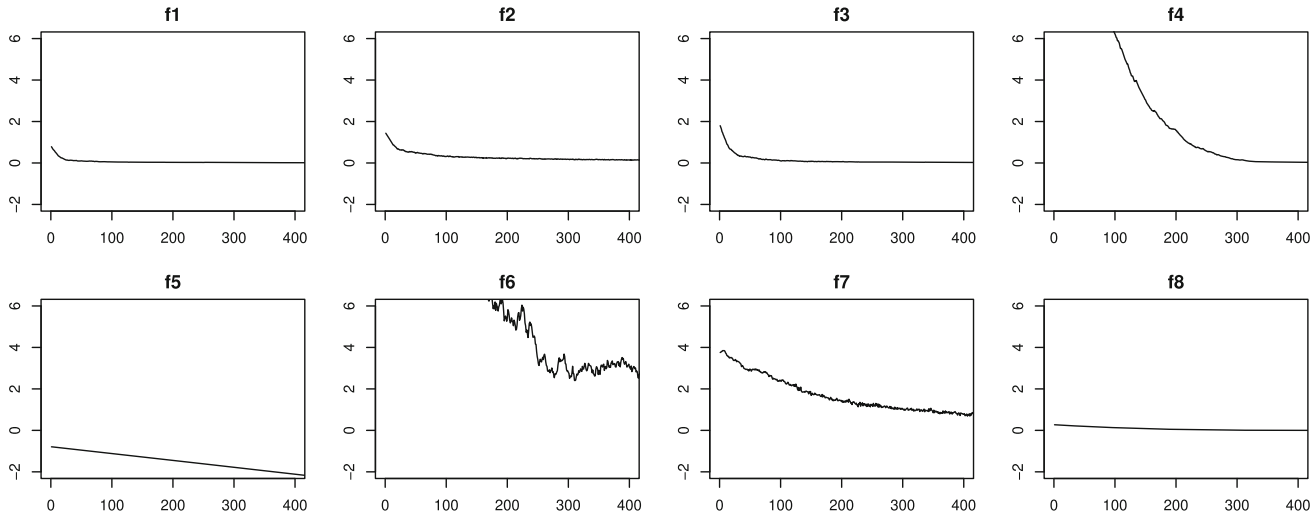

Fig. 6 Mean objective function for 20 executions for PbDRD with $p_m = 0.9$ for all problems f_1–f_8 considered and considering $s = 50$ agents. Function f_4, number of voting alternatives $k_\alpha = 8$. Parameters ω, $\phi_p = \phi_g$, and ω_G selected to obtain one of the best convergences

5.1 Local and global privacy

We have seen that when we use of a differentially private voting mechanism to aggregate the opinions of the agents, agent's use of local masking methods does not make a significant difference on the result.

The use of two different approaches (the local one and the global one) may be seen as a disproportionate zeal to ensure the privacy of agents. We consider that this is not the case. Local privacy is to ensure that the information supplied by an agent is not revealed if the system is compromised. Thus, in case of an architecture that is not trusted or that may be compromised, local protection is the best option. In this way, agent's privacy is not fully in hands of third parties.

Nevertheless, local protection may not be enough. An agent has only partial information on the whole system and of other's information. That is, the agent has its information on itself (e.g., its preferences) and partial information on the others. So, agents' knowledge on what information is sensitive is limited. The sensitivity of information can depend on the value itself. For example, reidentification from a transmitted value can only be done if this value is known to be unique. So, in general, agents may have not enough global knowledge to protect the data at an appropriate level. In contrast, the central authority can have this information available when gathers the preferences of all agents. By means of multiparty computation mechanisms (see Sect. 2), the votes of all agents can be aggregated, and using a differential privacy voting mechanism (or any other appropriate mechanism) a final decision is made. This global information can then be used to avoid other disclosure.

When databases are protected in a centralized environment, masking methods such as PRAM can be enough for data protection. This is so, because among different masking strategies we can select the ones that avoid reidentification doing appropriate risk analysis. The analysis of reidentification of a single record will take into account the other records in the database (and their masked counterparts). This is not possible by a single agent without information on the other's agent information. So, PRAM and randomized response may not be enough for ensuring agents privacy.

Naturally, more complex processes can be designed for agents so that they can collaborate (sharing information to trusted agents) so that decisions about their own protection is more learned.

Therefore, on the light of the results given and taking into account that the masking parameter does not affect so much the quality of the solution, we consider that the use of masking is a clear advantage.

5.2 Future work

One of the conclusions outlined above states that, in general, the use of privacy mechanisms does not prevent the system to converge into the optimal solution. While in a standard application, a slow convergence *just means* more time required. In the case of privacy solutions, we need to take into account that additional iterations can increase the risk of disclosure.

Differential privacy has nice properties with respect to composability. So, if two queries/functions are computed with ϵ_1 and ϵ_2 each, the privacy guarantee of both together (in a sequential composition) is $\epsilon_1 + \epsilon_2$. Randomization should be independent for each query/function. Therefore, we can consider a privacy budget and agents only informing of their position when they consider appropriate to do so. Here, with *appropriate to do so* we mean that agents need to take into

account their privacy budget as well as other considerations. Note here agents supply information related to their trajectory, so, voting for different directions is about releasing information that may lead to reidentification. Privacy budgets for agents may need to take into account the surface of the function being optimized.

We have worked with an homogeneous architecture where all agents have the same set of rules for updating their positions and for deciding where and what to communicate. In the real world, with agents representing individuals interests, different agents need to consider different privacy preferences and, therefore, different rules about transmitting information. We need to consider heterogeneous systems.

Funding Open access funding provided by Umea University. This study was partially funded by the Swedish Research Council (Vetenskapsrådet) (Grant Number VR 2016-03346), the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation, and the Spanish Ministry of Science and Innovation (PID2021-125962OB-C33).

Data availability Not applicable.

Declarations

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

1. Blanco-Justicia, A., Domingo-Ferrer, J., Martínez, S., Sánchez, D., Flanagan, A., Tan, K.E.: Achieving security and privacy in federated learning systems: survey, research challenges and future directions. Eng. Appl. Artif. Intell. **106**, 104468 (2021)
2. Bogdanov, D., Laur, S., Willemson, J.: Sharemind: a framework for fast privacy-preserving computations. In: Proceedings of ESORICS 2008, LNCS (2008)
3. Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H. B., Patel, S., Ramage, D., Segal, A., Seth, K.: Practical secure aggregation for privacy-preserving machine learning. In: Proceedings of CCS'17 (2017)
4. Duncan, G.T., Elliot, M., Salazar, J.J.: Statistical Confidentiality. Springer (2011)
5. Dwork, C.: Differential privacy. In: Proceedings of ICALP 2006. LNCS, vol. 4052, pp. 1–12 (2006)
6. Dwork, C.: Differential privacy: a survey of results. In: Proceedings of TAMC 2008. LNCS, vol. 4978, pp. 1–19 (2008)
7. Gouweleeuw, J.M., Kooiman, P., Willenborg, L.C.R.J., De Wolf, P.-P.: Post randomisation for statistical disclosure control: theory and implementation. J. Off. Stat. **14**(4), 463–478 (1998). Also as Research Paper No. 9731, Voorburg: Statistics Netherlands (1997)
8. Hundepool, A., Domingo-Ferrer, J., Franconi, L., Giessing, S., Nordholt, E.S., Spicer, K., de Wolf, P.-P.: Statistical Disclosure Control. Wiley (2012)
9. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: Proc. IEEE Int. Conf. Neural Networks, pp. 1942–1948 (1995)
10. Li, T., Sahu, A.K., Talwalkar, A., Smith, V.: Federated learning: challenges, methods, and future directions. IEEE Signal Process. Mag. **37**(3), 50–60 (2020)
11. Sakuma, J., Osame, T.: Recommendation with k-anonymized ratings. Trans. Data Privacy **11**, 47–60 (2018)
12. Sengupta, S., Basak, S., Peters, R.A., II.: Particle swarm optimization: a survey of historical and recent developments with hybridization perspectives. Mach. Learn. Knowl. Extr. **1**, 157–191 (2018)
13. Torra, V.: Data Privacy: Foundations, New Developments and the Big Data Challenge. Springer (2017)
14. Torra, V.: Random dictatorship for privacy-preserving social choice. Int. J. Inf. Secur. **19**, 537–545 (2020)
15. Vaidya, J., Clifton, C., Zhu, M.: Privacy Preserving Data Mining. Springer (2006)
16. Warner, S.L.: A survey technique for eliminating evasive answer bias. J. Am. Stat. Assoc. **60**(309), 63–69 (1965)
17. Yang, Q., Liu, Y., Chen, T., Tong, Y.: Federated machine learning: concept and applications. ACM Trans. Intell. Syst. Technol. **10**, 1–19 (2019). arxiv:1902.04885
18. Zhu, L., Liu, Z., Han, S.: Deep leakage from gradients. In: Proceedings of NeurIPS (2019)